2. 键控流水灯实验例程

2.1 MES50HP 开发板简介

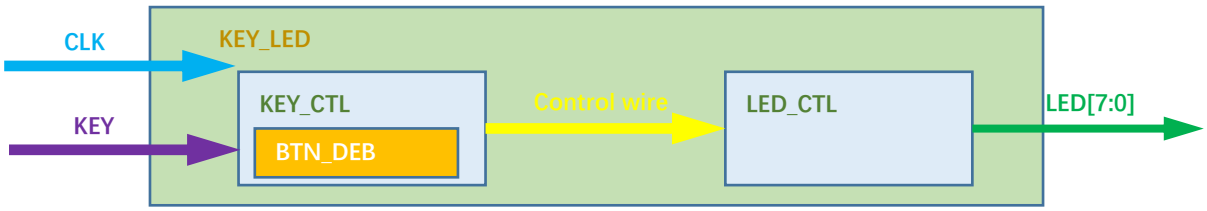
MES50HP 扩展底板提供了 8 个用户按键（USER_BUTTON1～8），按键按下时，IO 上的输入电压为低（详情请查看“MES50HP 开发板硬件使用手册”）。

2.2 实验目的

由 USER_BUTTON1 按键输入，切换 USER_LED1~ USER_LED8 的输出效果。

2.3 实验原理

实现框架如下：



（1）顶层实现按键切换 LED 的流水灯状态；

（2）需要设计一个输入控制模块及一个输出控制模块；

这个实验带大家将多个模块整合成为一个工程，涉及到的知识点有子模块设计、模块例化；子模块的设计主要是依据功能定位，确定输入输出，再做具体的设计；

模块例化方式如下：

```
module_name #（
        .PARAM      (  PARAM_SET )      // PARAM为例化模块的常量接口；PARAM_SET为常量赋值内容
) unint_name(       // module_name 为例化module名；unint_name为例化后单元名称
        .port       (  signal       )       // port为例化模块中的管脚；signal为当前模块的信号
);
```

2.3.1按键控制模块功能

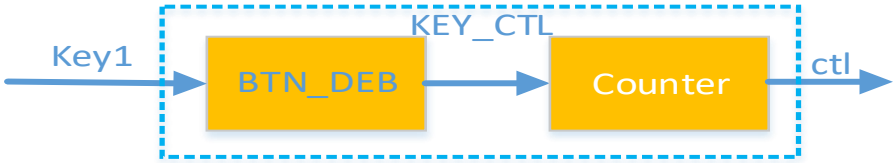
接收按键输入信号。统计按键按下次数，由于流水灯模式是 3 种，计数统计范围是 0～2 循环，将计数结果传递给 LED 控制模块；

根据需求输入信号有：时钟，按键；输出信号有：流水灯控制信号；

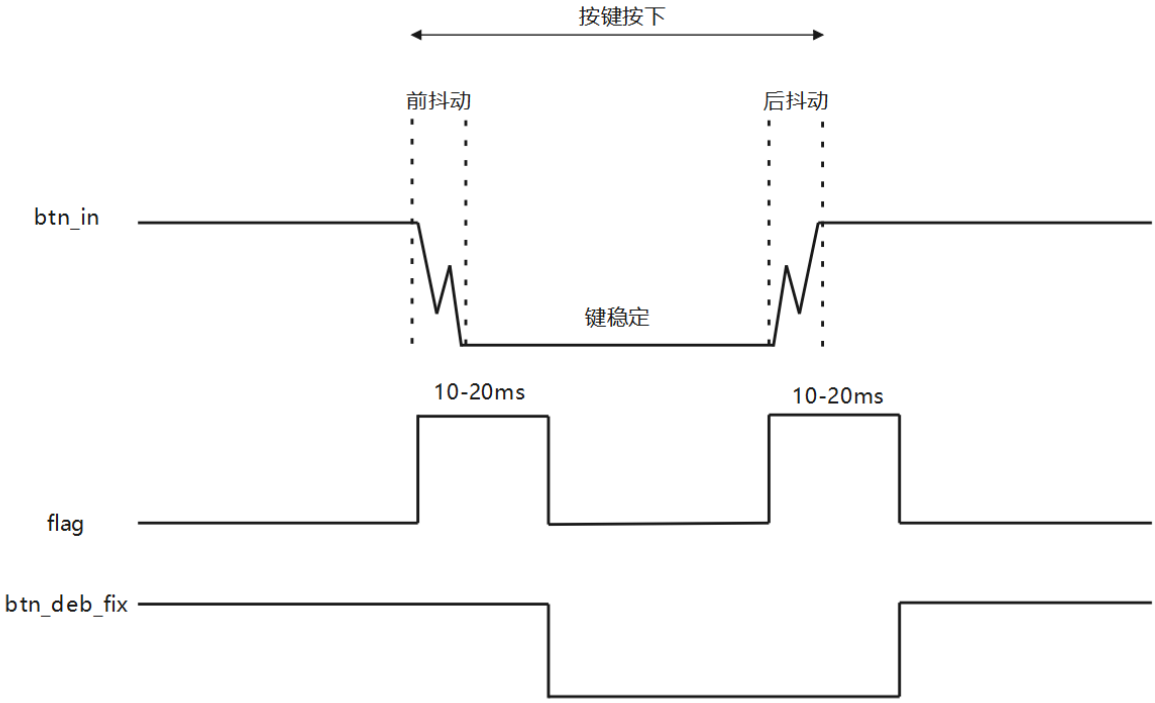
内部功能处理：

<1>内部需要对按键信号做消抖处理；

<2>按键触发计数器（计数值输出）改变继而调整流水灯的状态；



2.3.2按键消抖模块



前后抖动时间约为 5～10ms，取按键抖动区间开始标识，持续 10-20ms 后标识归零，在抖动区间内输出保持，非消抖区间，按键状态输出。

2.3.3LED 控制模块功能

3 种流水灯模式有按键传递过来的计数控制切换，每一个 LED 的显示状态完整后进入下一模式初始化。根据需求可得到如下信息：

输入信号：时钟，流水灯模式控制信号； 出信号：8bit 位宽的 LED 控制信号；

功能处理注意事项：流水灯状态切换点，不同状态的切换时如何初始化；

2.4 实验源码设计

2.4.1顶层文件源码

```
1   `timescale 1ns / 1ps
2
3   `define UD #1
4   module key_led_top(
5        input       clk,
6        input       key,
7        output [7:0]led
8   );
9
10      wire [1:0] ctrl;
11
12      key_ctl key_ctl(
13          .clk       (  clk  ),//input       clk,
14          .key       (  key  ),//input       key,
15          .ctrl      (  ctrl  )//output [1:0]ctrl
16      );
17
18      led u_led(
19          .clk   (  clk   ),//input       clk,
20          .ctrl  (  ctrl  ),//input [1:0] ctrl,
21
22          .led   (  led   ) //output[7:0] led
23      );
24
25  endmodule
```

2.4.2按键控制模块

```
`timescale 1ns / 1ps
`define UD #1
module key_ctl(
    input           clk,
    input           key,

    output     [1:0] ctrl
);

    wire btn_deb;
    // 按键消抖
    btn_deb#(
        .BTN_WIDTH   (  4'd1     ) //parameter                 BTN_WIDTH = 4'd8
    ) U_btn_deb
    (
        .clk         (  clk      ),//input                      clk,
        .btn_in      (  key      ),//input      [BTN_WIDTH-1:0] btn_in,

        .btn_deb     (  btn_deb  ) //output reg [BTN_WIDTH-1:0]btn_deb
    );

    reg btn_deb_1d;
    always @(posedge clk)
    begin
        btn_deb_1d <= `UD btn_deb; //get the btn_deb delay one clock cycle
    End

//下降沿获取方式：前一个时钟周期为高电平，当前时钟周期为低电平；
// 故而将按键消抖后的信号打一拍（保持上一时钟周期的状态）
//                ____________
//  sig                       |____________
//                _______________
//sig_reg                        |________
//                             __
//falling         ____________|  |________

    reg [1:0]  key_push_cnt=2'd0;
    always @(posedge clk)
    begin
        if(~btn_deb & btn_deb_1d) //get he falling edge of btn_deb
        begin
            key_push_cnt <= `UD key_push_cnt + 2'd1;
         end
    end

    assign ctrl = key_push_cnt;

endmodule
```

2.4.3按键消抖模块

```
1   `timescale 1ns / 1ps
2   `define UD #1
3   module btn_deb_fix#(
4       parameter                   BTN_WIDTH = 4'd8,
5       parameter                   BTN_DELAY = 20'h7_ffff
6   )
7   (
8       input                       clk,    //
9       input       [BTN_WIDTH-1:0] btn_in,
10
11      output reg [BTN_WIDTH-1:0] btn_deb_fix
12  );
13      //16'h3ad43;
14      reg [17:0]              cnt[BTN_WIDTH-1:0];
15      reg [BTN_WIDTH-1:0] flag;
16
17      reg [BTN_WIDTH-1:0] btn_in_reg;

18      always @(posedge clk)
19      begin
20        btn_in_reg <= `UD btn_in;
21      end
22
23      genvar i;
24      generate
25      begin
26          for(i=0;i<BTN_WIDTH;i=i+1)
27          begin
28              always @(posedge clk)
29              begin
30                if (btn_in_reg[i] ^ btn_in[i]) //取按键边沿开始抖动区间标识
31                  flag[i] <= `UD 1'b1;
32                else if (cnt[i]== BTN_DELAY)      //持续 20ms 后归零
33                  flag[i] <= `UD 1'b0;
34                  else
35                      flag[i] <= `UD flag[i];
36              end
37
38              always @(posedge clk)
39              begin
40                if(cnt[i]== BTN_DELAY)          //计数 20ms 时归零
41                  cnt[i] <= `UD 18'd0;
42                else if(flag[i])                //抖动区间有效时计数
43                  cnt[i] <= `UD cnt[i] + 1'b1;
44                else                                //非抖动区间保持 0
45                  cnt[i] <= `UD 18'd0;
46              end
47
48              always @(posedge clk)
49              begin
50                if(flag[i])                     //抖动区间，消抖输出保持
51                  btn_deb_fix[i] <= `UD btn_deb_fix[i];
52                else                                //非抖动区间，按键状态传递到消抖输出
53                  btn_deb_fix[i] <= `UD btn_in[i];
54              end
55          end
56      endgenerate
```

2.4.4LED 控制模块

```
1   `timescale 1ns / 1ps
2   `define UD #1
3   module led(
4       input         clk,
5       input  [1:0]  ctrl,
6       output [7:0]  led
7   );
8
9       reg [24:0] led_light_cnt = 25'd0;
10      reg [ 7:0] led_status = 8'b1000_0000;
11
12      //  time counter
13      always @(posedge clk)
14      begin
15          if(led_light_cnt == 25'd19_999_999)
16              led_light_cnt <= `UD 25'd0;
17          else
18              led_light_cnt <= `UD led_light_cnt + 25'd1;
19      end
20
21      reg [1:0] ctrl_1d;     //保存上一个 led 状态周期的 ctrl 值
22      always @(posedge clk)
23      begin
24          if(led_light_cnt == 25'd19_999_999)
25              ctrl_1d <= ctrl;//此处设计能保证状态切换时，从 0 时刻开始下一次流水状态
26      end
27
28      // led status change
29      always @(posedge clk)
30      begin
31          if(led_light_cnt == 25'd19_999_999)//0.5s 周期
32          begin
33              case(ctrl)
34                  2'd0 :  //从高位到低位的 led 流水灯
35                  begin
36                      if(ctrl_1d != ctrl)
37                          led_status <= `UD 8'b1000_0000;
38                      else
39                          led_status <= `UD {led_status[0],led_status[7:1]};
40                  end
41                  2'd1 :  //从地位到高位的 led 流水灯
42                  begin
43                      if(ctrl_1d != ctrl)
44                          led_status <= `UD 8'b0000_0001;
45                      else
46                          led_status <= `UD {led_status[6:0],led_status[7]};
47                  end
48
```

```
49            2'd2 :  //从低位到高位增加亮灯的个数
50                begin
51                    if(ctrl_1d != ctrl || led_status == 8'b1111_1111)
52                        led_status <= `UD 8'b0000_0000;
53                    else
54                        led_status <= `UD {led_status[6:0],1'b1};
55                end
56                2'd3 :  //从高位到低位增加灭灯的个数
57                begin
58                    if(ctrl_1d != ctrl || led_status == 8'b0000_0000)
59                        led_status <= `UD 8'b1111_1111;
60                    else
61                        led_status <= `UD {1'b0,led_status[7:1]};
62                end
63             endcase
64         end
65     end
66
67     assign led = led_status;
68
69 endmodule
70
```

2.5 实验现象

每按下一次 KEY1，LED 灯状态切换一次，总共三种 LED 模式供循环切换；

LED 模式一：从高位到低位的 LED 流水灯；

LED 模式二：隔一亮一交替点亮；

LED 模式三：从高位到低位暗灯流水；